RECEIVED
CENTRAL FAX CENTER
OCT 10 2006

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Docket No. 13270US01

IN THE APPLICATION OF:

D. Hylands et al.

SERIAL NO.: 10/032,667

FILED: October 24, 2001

FOR: TRANSFERRING DATA ALONG WITH CODE FOR PROGRAM OVERLAYS

ART UNIT: 2194

EXAMINER: Andy Ho

Conf. No.: 7248

CERTIFICATE OF FACSIMILE

I hereby certify that this correspondence is being transmitted via facsimile to the United States Patent Office on October 10, 2006.

John A. Wiberg
Reg. No. 44,401

BRIEF ON APPEAL

Mail Stop: Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This is an appeal from an Office Action dated March 7, 2006, in which claims 1-28 were finally rejected.

REAL PARTY IN INTEREST

Broadcom Corporation, a corporation organized under the laws of the state of California, and having a place of business at 16215 Alton Parkway, Irvine, California 92618-3616, has acquired the entire right, title and interest in and to the invention, the application, and any and all patents to be obtained therefor, as set forth in the Assignment recorded at Reel 012614, Frame 0097 in the PTO assignment search room.

RECEIVED
CENTRAL FAX CENTER

OCT 1 0 2006

RELATED APPEALS AND INTERFERENCES

There currently are no appeals pending regarding related applications.

STATUS OF THE CLAIMS

Claims 1-28 are pending in the present application. Pending claims 1-28 stand rejected and are the subject of this appeal.

STATUS OF THE AMENDMENTS

An Amendment is submitted herewith to correct Figures 6B and 7. The changes merely correct errors in the formal drawings submitted on April 8, 2002. The Figure 6B as amended reflects the informal Figure 6B submitted with the original application on October 24, 2001. The Figure 7 as amended reflects the informal Figure 7 submitted with the original application. The amended Figures 6B and 7 conform to the associated text in the Specification. Therefore Figures 6B and 7 as amended are supported by the application as originally filed. Because this Amendment introduces no new issues, Appellant requests entry of the Amendment prior to consideration of the appeal.

SUMMARY OF CLAIMED SUBJECT MATTER

Claim 1 is directed to a method for generating program overlays from a sequence of program code. Each overlay has a set of code and related data contained therein. The overlays are transferred via an overlay manager from a storage area to a receiving area for processing. In a step (a), the sequence of program code is broken into a set of segments, each segment containing a certain amount of related code for processing. In a step (b), a code segment in the set is run through a linker device. In a step (c), the code segment and related data segment produced by the linker device is extracted, each associated pair of code and data segments representing an overlay. In a step (d), it is checked whether more segments exist in the set. If yes, then return to step (b), else

proceed to step (e). In a step (e), the associated code is concatenated into paired sets which can be referenced by the overlay manager.

The invention of claim 1 is illustratively described in the Specification of the present application at, for example, paragraph 52, referring to Figure 6A. At step 602, the code is broken into a set of segments. At step 604, a code segment is run through a linker device. At step 606, the code segment and related data segment produced bt the linker are extracted. At step 608, it is determined whether more code segments exist in the set of segments. If so, the process returns to step 604. If not, the process proceeds to step 610, wherein the code segments and data segments are concatenated together into paired sets that can be referenced by the overlay manager. The invention of claim 1 is also described in other parts of the application, such as in the Summary of the Invention section.

Claims 2-11 are dependent upon claim 1.

Claim 12 is directed to a method for generating program overlays from a sequence of program code. The program code has common code and code to be overlaid. Each overlay has a set of code and related data contained therein. The overlays are transferred via an overlay manager from a storage area to a receiving area for processing, Pursuant to the method, in a step (a), a memory segment is reserved in the receiving area to hold overlaid code and data. In a step (b), the sequence of code to be overlaid is broken into a set of segments, each segment containing a certain amount of related code for processing and being sized to fit in the reserved memory segment. In a step (c), stubs are created for each code segment. The stubs represent entry points for functions within each code segment. In a step (d), the common code is linked along with the stubs for each code segment. In a step (e), symbols are imported from the common code and the next individual code segment in the set of segments is linked to produce an image. In a step (f), overlay code and data are extracted from the image produced in step (e). In a step (g), it is determined whether more segments exist in the set of segments. If yes, then return to step (e), else proceed to step (h). In a step (h), the associated code and data segments are concatenated into paired sets which can be referenced by the overlay manager.

The invention of claim 12 is illustratively described in the Specification of the present application at, for example, paragraphs 53-57, referring to Figure 6B. At step

652, a memory segment is reserved in the receiving area to hold overlaid code and data. At step 654, the sequence of code to be overlaid is broken into a set of segments, each segment containing a certain amount of related code for processing and being sized to fit in the reserved memory segment. At step 656, stubs are created for each code segment. The stubs represent entry points for functions within each code segment. At step 658, the common code is linked along with the stubs for each code segment. At step 660, symbols are imported from the common code and the next individual code segment in the set of segments is linked to produce an image. At step 662, overlay code and data are extracted from the image produced in step 660. At step 664, it is determined whether more segments exist in the set of segments. If so, the process returns to step 660. Otherwise, the process proceeds to step 667, where the associated code and data segments are concatenated into paired sets which can be referenced by the overlay manager. The invention of claim 12 is also described in other parts of the application, such as in the Summary of the Invention section.

Claims 13-18 are dependent upon claim 12.

Claim 19 is directed to a method for generating program overlays from a sequence of program code. The program code has common code area and overlay code area. Each overlay has a set of code and related data contained therein. The overlays are transferred via an overlay manager from a storage area to a receiving area for processing. Pursuant to the method, the overlay code area is analyzed and the function entry points for each overlay are determined. An overlay control file is created for each overlay. The overlay control file describes each pair of code and data associated with each overlay. A wrapper file is generated from the overlay control file. A linker command file for the common area is created. A linker command file for the overlay area is created. An initialization is performed for the overlay. A common image is created for the code and data. Then overlay sections are produced from the image. An overlay sections file is produced. A load command file is also produced. The command file will load the overlay sections file into the appropriate receiving area.

The invention of claim 19 is illustratively described in the Specification of the present application at, for example, paragraphs 71-87, referring to Figure 7. At step 702, the function entry points for each overlay are determined. At step 704, an overlay control file is created for each overlay. The overlay control file describes each pair of

code and data associated with each overlay. At step 706, a wrapper file is generated from the overlay control file. At step 708, a linker command file for the common area is created. At step 710, a linker command file for the overlay area is created. An initialization is performed for the overlay, as described in paragraph 87 of the Specification. At step 714, a common image is created for the code and data. Then, at step 716, overlay sections are produced from the image. At step 718, an overlay sections file is produced. At step 718, a load command file is also produced. The command file will load the overlay sections file into the appropriate receiving area, as described in paragraph 86. The invention of claim 19 is also described in other parts of the application, such as in the Summary of the Invention section.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

I. Claims 1-28 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Admitted Prior Art in view of U.S. Patent 6,209,061 issued to Marvin D. Nelson, et al.

RECEIVED
CENTRAL FAX CENTER
OCT 10 2006

ARGUMENT

I. Claims 1-18 are not obvious under 35 U.S.C. § 103(a) in view of the Admitted Prior Art and Nelson (U.S. Patent 6,209,061).

Claims 1-18 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over the Admitted Prior Art (APA) in view of Nelson (U.S. Patent 6,209,061). 35 U.S.C. 103(a) states:

> A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.[1]

The Supreme Court laid out the standard of patentability to be applied in obviousness rejections in *Graham v. John Deere,* stating:

> Under § 103, the scope and content of the prior art are to be determined; differences between the prior art and the claims at issue are to be ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background, the obviousness or nonobviousness of the subject matter is determined.[2]

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art.[3]

Claim 1 is directed to "a method for generating program overlays from a sequence of program code" and includes the following steps:

> (a) breaking the sequence of program code into a set of segments, wherein each segment contains a certain amount of related code for processing;
> (b) running a code segment in the set through a linker device;
> (c) extracting the code segment and related data segment produced by the linker device, with each associated pair of code and data segments representing an overlay;
> (d) checking if more segments exist in the set, if yes, then return to step (b), else proceed to step (e); and
> (e) concatenating the associated code and data segments into paired sets which can be referenced by the overlay manager.

---

[1] 35 U.S.C. § 103(a).
[2] *Graham v. John Deere*, 383 U.S. 1, 148 USPQ 459 (1966).
[3] *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974).

Thus, steps (b) through (d) allow for the creation of multiple overlays, each comprising code and the data associated with that code, by utilizing multiple passes of the linker on multiple segments of code. The Examiner acknowledges that the Admitted Prior Art (APA) does not teach the use of multiple passes of a linker on the segments of code.[4] The Examiner asserts that this aspect of the present invention is taught by Nelson.[5] Applicant respectfully submits that Nelson says nothing about the process of generating overlays. Rather, Nelson merely talks about a method of using overlays.[6] Therefore, Applicant submits that Nelson does not teach any of the steps of claim 1. In particular, Nelson does not teach utilizing multiple passes of a linker on multiple segments of code. The Examiner apparently asserts that this aspect of the present invention is disclosed by Nelson at col. 3, lines 9-19.[7] Again, Applicant submits that this excerpt (nor any other excerpt) of Nelson does not even address a method of generating overlays. In particular, this excerpt (nor any other excerpt) does not say anything about using a linker, let alone anything about using multiple passes of a linker to generate overlays. Furthermore the overlays used in Nelson contain only data and parameters.[8] This is in contrast to the overlays generated by the method of claim 1, wherein the overlays comprise both code and data.

The Examiner also acknowledges that the Admitted Prior Art (APA) does not teach concatenating step (e).[9] The Examiner asserts that this aspect of the present invention is taught by Nelson at col. 3, lines 10-19.[10] The Examiner asserts that the fact that "the overlay memory 23 can be segmented into a number of memory regions, with each region defined by a base pointer value" somehow teaches concatenating together associated code and data segments into paired sets.[11] Applicant submits again that the overlays of Nelson consist only of data and parameters, rather than code and data.[12]

---

[4] Office Action dated March 7, 2006, page 3.
[5] *Id.*
[6] See, e.g., col. 3, lines 5-19, of Nelson (U.S. Patent 6,209,061).
[7] Office Action dated March 7, 2006, page 3.
[8] See col. 3, lines 6-9 of Nelson.
[9] Office Action dated March 7, 2006, page 3.
[10] *Id.*, pages 3-4.
[11] *Id.* (quoting Nelson, col. 3, lines 10-12).
[12] See, e.g., col. 3, lines 5-19, of Nelson (U.S. Patent 6,209,061).

Therefore, Nelson cannot teach concatenating step (d). Furthermore, Nelson only talks about using overlays, and does not describe a method of generating said overlays.

In the "Response to Arguments" section of the 3/7/06 Office Action, the Examiner notes that Appellant argued in a previous Response that Nelson is not directed to a method of generating overlays, but rather to a method of *using* overlays.[13] The Examiner responds by saying that APA was used to teach this limitation, not Nelson.[14] However, Appellant would point out that *all* of claim 1 is directed to a method of generating overlays, and the Examiner does in fact rely on Nelson for teaching some of the elements of claim 1. In particular, the Examiner relies on Nelson for teaching the use of multiple passes of a linker on multiple segments of code to generate overlays, and for teaching concatenating the associated code and data segments, extracted in step (c), into paired sets which can be referenced by the overlay manager.[15] But since Nelson is not even directed to a method of generating overlays, but instead to a method of using overlays, Nelson cannot and does not teach these aspects of claim 1, as is pointed out above.

Also in the "Response to Arguments" section of the 3/7/06 Office Action, the Examiner asserts that Appellant argued in a previous Response that Nelson does not teach a linker, and then asserts that the APA was used to teach this limitation, not Nelson.[16] Firstly, to clarify, Appellant's argument was that Nelson does not teach using multiple passes of a linker to generate overlays.[17] And the Examiner does indeed acknowledge that the APA does not teach this aspect of claim 1, specifically in the third paragraph of page 3 of the 3/7/06 Office Action, and then proceeds in the next paragraph to argue that this aspect of claim 1 is taught by Nelson.[18] As is pointed out above, Nelson does not, in fact, teach this limitation.

For at least the reasons outlined above, Applicant submits that claim 1, and claims 2-11 depending therefrom, are not taught by the combination of the admitted prior art and Nelson.

---

[13] Office Action dated March 7, 2006, page 11.
[14] *Id.*
[15] *Id.*, page 3.
[16] Office Action dated March 7, 2006, page 11.
[17] Amendment mailed on December 13, 2005, page 7, last paragraph.
[18] Office Action dated March 7, 2006, page 3, 3rd and 4th paragraphs.

Claim 12 contains some limitations that are similar to claim 1. For example, steps (e) through (g) allow for the creation of multiple overlays, each comprising code and the data associated with that code, by utilizing multiple passes of the linker on multiple segments of code. The Examiner acknowledges that the Admitted Prior Art (APA) does not teach the use of multiple passes of a linker on the segments of code, nor concatenating step (h).[19] The Examiner asserts that these aspects of the present invention are taught by Nelson.[20] Applicant respectfully disagrees and submits that claim 12 is allowable for at least the reasons set out above with respect to claim 1. Thus, Applicant submits that claim 12, and claims 13-18 depending therefrom, are not taught by the combination of the admitted prior art and Nelson.

II. Claims 19-28 are not obvious under 35 U.S.C. § 103(a) in view of the Admitted Prior Art and Nelson (U.S. Patent 6,209,061).

Claim 19 is directed to "a method for generating program overlays from a sequence of program code" and includes the following steps:

(a) analyzing the overlay code area and determining the function entry points for each overlay;
(b) creating an overlay control file for each overlay, whereby the overlay control file describes each pair of code and data associated with each overlay;
(c) generating a wrapper file from the overlay control file;
(d) creating a linker command file for the common area;
(e) creating a linker command file for the overlay area;
(f) performing an initialization for the overlay;
(g) creating a common image for the code and data;
(h) producing overlay sections from the image;
(i) producing an overlay sections file; and
(j) producing a load command file, whereby the command file will load the overlay sections file into the appropriate receiving area.

Like claims 1 and 12, claim 19 was rejected over the combination of the APA and Nelson. The Examiner acknowledges on page 8 of the Final Office Action that the APA does not teach step (b): "creating an overlay control file for each overlay, whereby the overlay control file describes each pair of code and data associated with each overlay."

[19] *Id.*, pages 6-7.
[20] *Id.*, page 7.

The Examiner asserts that step (b) is taught by Nelson at column 4, lines 5-10.[21] Column 4, lines 5-10, of Nelson refer to an overlay memory controller 24, which holds the respective base pointers of the subregions of the overlay memory 23. Applicant submits that this overlay memory controller 24 does not constitute an overlay control file for each overlay, and in particular, an overlay control file that describes each pair of code and data associated with each overlay, as called for in claim 19. Applicant further points out that the overlays of Nelson consist of data only. See column 3, lines 6-9, which say, "Overlay memory 23 is a relatively small, high-speed memory which provides storage for data and parameters that are temporarily required for a program 26 that is currently running on CPU 12." In contrast, the overlays of the present invention comprise both code and data. In the "Response to Arguments" section of the 3/7/07 Office Action, the Examiner asserts that APA was used to teach this limitation.[22] However, step (b) claims "creating an overlay control file" that "describes each pair of code and data associated with each overlay." The Examiner explicitly acknowledeges that this step is not taught by the APA.[23] Therefore, Applicant submits that step (b) is not taught by the APA nor by Nelson.

Further regarding claim 19, the Examiner asserts that step (c), "generating a wrapper file from the overlay control file," is taught by the APA at paragraphs 13 and 15.[24] Applicant submits that while these paragraphs refer generically to a wrapper, they do not describe generating a wrapper file from an overlay control file.

The Examiner further asserts regarding claim 19 that step (d), "creating a linker command file for the common area," is taught by the APA at paragraph 13.[25] Applicant submits that paragraph 13 of the present application says nothing about a linker command file and, more particularly, says nothing about a linker command file for the common area.

The Examiner further asserts regarding claim 19 that step (e), "creating a linker command file for the overlay area," is taught by the APA at paragraph 13.[26] Applicant

[21] *Id.*, page 9, lines 14-20.
[22] *Id.*, page 11, 2nd paragraph.
[23] *Id.*, 2006, page 8.
[24] *Id.*
[25] *Id.*
[26] *Id.*

submits that paragraph 13 of the present application says nothing about a linker command file and, more particularly, says nothing about a linker command file for the overlay area.

Further regarding claim 19, the Examiner asserts that step (f), "performing an initialization for the overlay," is taught by the APA at paragraph 15.[27] Applicant submits that paragraph 15 says nothing about performing an initialization for an overlay. Applicant further submits that performing step (f) is performed in the generating of the overlays, as indicated in the preamble of claim 19. Paragraph 15 talks about using overlays, but does not say anything about the process of generating said overlays.

The Examiner further asserts regarding claim 19 that step (j), "producing a load command file, whereby the command file will load the overlay sections file into the appropriate receiving area," is taught by the APA at paragraph 16.[28] Applicant submits that paragraph 16 of the present application says nothing about a load command file.

For at least the reasons outlined above, Applicant submits that claim 19, and claims 20-28 depending therefrom, are not taught by the combination of the admitted prior art and Nelson.

[27] *Id.*
[28] *Id.*

III. Conclusion

For at least the foregoing reasons, Appellant submits that claims 1-28 are not obvious in view of the admitted prior art and Nelson. Reversal of the Examiner's rejection and issuance of a patent on the application are therefore requested.

The Commissioner is hereby authorized to charge $500 (to cover the Brief on Appeal Fee of $500) and any additional fees or credit any overpayment to the deposit account of McAndrews, Held & Malloy, Account No. 13-0017.

Dated: 10/10/06

Respectfully submitted,

John A Wiberg
Registration No. 44,401
Attorney for appellant

McANDREWS, HELD & MALLOY, LTD.
500 West Madison Street, 34th Floor
Chicago, IL 60661
Telephone: (312) 775-8000
Facsimile: (312) 775-8100

CLAIMS APPENDIX (37 C.F.R. § 41.37(c)(1)(viii))

The following claims are involved in this appeal:

1. A method for generating program overlays from a sequence of program code, each overlay having a set of code and related data contained therein, the overlays being transferred via an overlay manager from a storage area to a receiving area for processing, the method comprising the steps of:

(a) breaking the sequence of program code into a set of segments, wherein each segment contains a certain amount of related code for processing;

(b) running a code segment in the set through a linker device;

(c) extracting the code segment and related data segment produced by the linker device, with each associated pair of code and data segments representing an overlay;

(d) checking if more segments exist in the set, if yes, then return to step (b), else proceed to step (e); and

(e) concatenating the associated code and data segments into paired sets which can be referenced by the overlay manager.

2. The method according to Claim 1, wherein the step of breaking the sequence of program code into a set of segments includes dividing the code into a common code area, and an overlay code area.

3. The method according to Claim 1, wherein the step of breaking the sequence of program code into a set of segments includes sizing the program code segments so that they will fit within the receiving area.

4. The method according to Claim 1, wherein the steps further include creating stubs for referencing each function in each program code segment, the stubs being stored in the receiving area.

5. The method according to Claim 4, wherein the steps further include generating an overlay table to be used in facilitating transfer of the overlays from the storage area to the receiving area, the overlay table being stored in the receiving area.

6. The method according to Claim 1, wherein the storage area includes an external storage means.

7. The method according to Claim 1, wherein the storage area includes memory associated with a low-MIPS processing device.

8. The method according to Claim 1, wherein the receiving area includes memory associated with a high-MIPS processing device.

9. The method according to Claim 8, wherein the high-MIPS processing device includes a digital signal processor.

10. The method according to Claim 1, wherein after the concatenating step, the information is converted into a form usable by a processor.

11. The method according to Claim 10, wherein the form includes a source file of a high-level programming language.

12. A method for generating program overlays from a sequence of program code, the program code having common code and code to be overlaid, each overlay having a set of code and related data contained therein, the overlays being transferred via an overlay manager from a storage area to a receiving area for processing, the method comprising the steps of:

(a) reserving a memory segment in the receiving area to hold overlaid code and data;

(b) breaking the sequence of code to be overlaid into a set of segments, wherein each segment contains a certain amount of related code for processing, and each segment is sized to fit in the reserved memory segment;

(c) creating stubs for each code segment, whereby the stubs represent entry points for functions within each code segment;

(d) linking the common code along with the stubs for each code segment;

(e) importing symbols from the common code and linking the next individual code segment in the set of segments to produce an image;

(f) extracting overlay code and data from the image produced in step (e);

(g) checking if more segments exist in the set, if yes, then return to step (e), else proceed to step (h); and

(h) concatenating the associated code and data segments into paired sets which can be referenced by the overlay manager.

13. The method according to Claim 12, wherein the storage area includes an external storage means.

14. The method according to Claim 12, wherein the storage area includes memory associated with a low-MIPS processing device.

15. The method according to Claim 12, wherein the receiving area includes memory associate with a high-MIPS processing device.

16. The method according to Claim 15, wherein the high-MIPS processing device includes a digital signal processor.

17. The method according to Claim 12, wherein after the concatenating step, the information is converted into a form usable by a processor.

18. The method according to Claim 17, wherein the form includes a source file of a high-level programming language.

19. A method for generating program overlays from a sequence of program code, the program code having common code area and overlay code area, each overlay having a set of code and related data contained therein, the overlays being transferred via an overlay manager from a storage area to a receiving area for processing, the method comprising the steps of:

(a) analyzing the overlay code area and determining the function entry points for each overlay;

(b) creating an overlay control file for each overlay, whereby the overlay control file describes each pair of code and data associated with each overlay;

(c) generating a wrapper file from the overlay control file;

(d) creating a linker command file for the common area;

(e) creating a linker command file for the overlay area;

(f) performing an initialization for the overlay;

(g) creating a common image for the code and data;

(h) producing overlay sections from the image;

(i) producing an overlay sections file; and

(j) producing a load command file, whereby the command file will load the overlay sections file into the appropriate receiving area.

20. The method of Claim 19, wherein the step of producing overlay sections from the image includes the following steps:

(a) creating a copy of the common image, whereby the entry point symbols are removed from the particular overlay to be built;

(b) linking together an image for a particular overlay to form an overlay image file; and

(c) extracting the code and data sections from the overlay image file.

21. The method of Claim 19, wherein the step of generating a wrapper file reads the overlay control file and generates wrapper functions for each function described therein

22. The method of Claim 21, wherein the wrapper file includes:

an overlay descriptor, which resides in common data and contains information about the overlay;

the wrapper function, which is the entry point to the overlay function; and

a fault function, which causes the overlay code and data sections to be paged from the storage area to the receiving area.

23. The method according to Claim 19, wherein the storage area includes an external storage means.

24. The method according to Claim 19, wherein the storage area includes memory associated with a low-MIPS processing device.

25. The method according to Claim 19, wherein the receiving area includes memory associated with a high-MIPS processing device.

26. The method according to Claim 25, wherein the high-MIPS processing device includes a digital signal processor.

27. The method according to Claim 19, wherein after the concatenating step, the information is converted into a form usable by a processor.

28. The method according to Claim 27, wherein the form includes a source file of a high-level programming language.

EVIDENCE APPENDIX (37 C.F.R. § 41.37(c)(1)(ix))

None.

RELATED PROCEEDINGS APPENDIX (37 C.F.R. § 41.37(c)(1)(x))

None.